AMENDMENTS TO THE CLAIMS

Claims 1-38 are currently pending in the Application. Claims 1, 18, and 27-30 are currently amended to clarify the claimed invention as embodied in these claims, without acquiescence or prejudice to pursue the original claims in a related application. A complete listing of the current pending claims is provided below and supersedes all previous claims listing(s). No new matter has been added.

1. (Currently Amended) A method for debugging an electronic design comprising both an HDL portion and a general programming language portion, comprising:

interrupting a simulator that operates upon the HDL portion of the electronic design to allow for debugging, ~~of the HDL portion, the simulator interrupted~~ by an external debugger, ~~to debug~~ the general programming language portion of the electronic design, wherein the simulator receives a plurality of simulator requests for simulation of the HDL portion from a programming interface when the simulator is not interrupted;

handling [[a]]one of the plurality of simulator requests, which is identified by and sent to the external debugger by an programming interface, with the external debugger for the simulator that is interrupted, the external debugger calling a request processing module ~~function~~ at the simulator to process the one of the plurality of simulator requests, the plurality of simulator requests for the simulation of the HDL portion;

executing a process of the request processing module ~~function~~ at the simulator to respond to the one of the plurality of simulator requests, wherein the act of executing the process of the request processing module is performed by a processor; [[and]]

providing continued access to the HDL portion while debugging, by using the external debugger, the general programming portion; and

generating debug results based upon executing the request processing function and storing the debug results in a computer-readable storage medium or a storage device or displaying the debug results on a display apparatus.

2. (Original) The method of claim 1 in which the simulator request accesses a portion of the HDL portion.

3. (Original) The method of claim 2 in which the simulator request accesses HDL signal values.

4. (Original) The method of claim 2 in which the simulator request accesses HDL design hierarchy.

5. (Original) The method of claim 1 in which the simulator request operates simulator functionality.

6. (Original) The method of claim 1 in which the general programming language portion comprises C, C++, or SystemC code.

7. (Original) The method of claim 1 in which the HDL portion comprises VHDL or Verilog.

8. (Original) The method of claim 1 in which the action of having the external debugger call the request processing function is based upon recognition of a waiting simulator request.

9. (Original) The method of claim 8 in which recognition of the waiting simulator request is based upon a message sent to the external debugger.

10. (Original) The method of claim 8 in which recognition of the waiting simulator request is based upon a periodic check of a simulator request wait queue.

11. (Original) The method of claim 8 in which recognition of the waiting simulator request is based on whether a threshold number of simulator requests are waiting in a simulator request wait queue.

12. (Original) The method of claim 1 in which the simulator request is generated at a simulator GUI.

13. (Original) The method of claim 12 in which the response to the simulator request is displayed at the simulator GUI.

14. (Previously Presented) The method of claim 1, wherein the external debugger that calls the request processing function at the simulator is a gdb debugger.

15. (Original) The method of claim 1 in which the simulator request is routed through a debugger GUI for the external debugger.

16. (Original) The method of claim 1 in which the simulator request is directly routed to the external debugger.

17. (Original) The method of claim 1 in which the request processing function is set up ahead of time at the simulator to handle anticipated simulator requests.

18. (Currently Amended) A method for processing of a design that is based upon multiple programming languages, the design comprising a first language portion and a second language portion, the method comprising:

~~processing~~ performing a first process on the second language portion of the design to cause an interruption of processing for the first language portion of the design, wherein the processing for the first language portion is interrupted to process the second language portion, and the first language portion receives a plurality of requests for the processing for the first language portion from a programming interface when the first language portion is not interrupted;

determining whether there are one or more ~~waiting~~ of the plurality of requests, which are identified by and sent to a processing module to perform the first process on the second language portion by a programming interface, waiting for the processing of the first language portion and indicating a need for the processing module ~~of the~~

~~second language portion~~ to call a request processing function at the first language portion;

handling the one or more ~~waiting~~ of the plurality of requests, which are identified by and sent to the processing module by the programming interface, for the processing of the first language portion by having the processing module ~~of the second language portion~~ call [[a]]the request processing function at the first language portion that has been interrupted, at least one of the one or more ~~waiting~~ of the plurality of requests for the processing of the first language portion ~~causes the processing of the first language portion~~;

executing the request processing function at the first language portion to process the one or more ~~waiting~~ of the plurality of requests, wherein the act of executing the request processing function is performed by a processor; and

providing continued access to the first language portion while performing the first process on the second portion, by using the processing module, the second language portion; and

generating processing results based upon executing the request processing function and storing the processing results in a computer-readable storage medium or a storage device or displaying the processing results on a display apparatus.

19. (Original) The method of claim 18 in which the one or more waiting requests are for accessing data from the first language portion of the design.

20. (Original) The method of claim 18 in which the one or more waiting requests are for debugging the first language portion.

21. (Previously Presented) The method of claim 18, wherein the act of determining whether there are one or more waiting requests for processing of the first language portion is based upon a message sent to a debugger for the processing of the second language portion.

22. (Previously Presented) The method of claim 18, wherein the act of determining whether there are one or more waiting requests for processing of the first language portion is based a periodic check of a request wait queue for the first language portion.

23. (Previously Presented) The method of claim 18, wherein the act of determining whether there are one or more waiting requests for processing of the first language portion is based on whether a threshold number of simulator requests are waiting in a request wait queue.

24. (Previously Presented) The method of claim 18, wherein the request processing function is called by a gdb debugger.

25. (Original) The method of claim 18 in which processing the second language portion comprises debugging the second language portion.

26. (Original) The method of claim 18 in which the request processing function is set up ahead of time to handle anticipated requests.

27. (Currently Amended) A computer program product comprising a ~~volatile or non-volatile~~ computer ~~usable~~ readable storage medium having executable code, when executed by a computer, causes the computer to execute a process for debugging of an electronic design comprising both an HDL portion and a general programming language portion, the process comprising:

interrupting a simulator that operates upon the HDL portion of the electronic design to allow for debugging, ~~of the HDL portion, the simulator interrupted~~ by an external debugger, ~~to debut~~ the general programming language portion of the electronic design, wherein the simulator receives a plurality of simulator requests for simulation of the HDL portion from a programming interface when the simulator is not interrupted;

handling [[a]]one of the plurality of simulator requests, which is identified by and sent to the external debugger by a programming interface, with the external debugger for the simulator that is interrupted, the external debugger calling a request processing

module ~~function~~ at the simulator to process the one of the plurality of simulator requests, the plurality of simulator requests for simulation of the HDL portion;

executing a process of the request processing module ~~function~~ at the simulator to respond to the one of the plurality of simulator requests, wherein the act of executing the process of the request processing module is performed by a processor; [[and]]

providing continued access to the HDL portion while debugging, by using the external debugger, the general programming portion; and

generating debug results based upon executing the request processing function and storing the debug results in a computer-readable storage medium or a storage device or displaying the debug results on a display apparatus.

28. (Currently Amended) A system for debugging of an electronic design comprising both an HDL portion and a general programming language portion, comprising:

means for interrupting a simulator that operates upon the HDL portion of the electronic design to allow for debugging, ~~of the HDL portion, the simulator interrupted~~ by an external debugger, ~~to debug~~ the general programming language portion of the electronic design, wherein the simulator receives a plurality of simulator requests for simulation of the HDL portion from a programming interface when the simulator is not interrupted;

means for handling [[a]]one of the plurality of simulator requests, which is identified by and sent to the external debugger by a programming interface, with the external debugger for the simulator that is interrupted, the external debugger calling a request processing module ~~function~~ at the simulator to process the one of the plurality of simulator requests, the plurality of simulator request for simulation of the HDL portion;

means for executing a process of the request processing module ~~function~~ at the simulator to respond to one of the plurality of the simulator request, wherein the means for

executing the process of the request [[for]]processing module ~~function~~ comprises a processor; and

means for providing continued access to the HDL portion while debugging, by using the external debugger, the general programming portion; and

means for generating debug results based upon executing the request processing function and a computer readable storage medium or a storage device configured for storing the debug results or a display apparatus configured for displaying the debug results ~~in a computer-readable medium~~.

29. (Currently Amended) A computer program product comprising a ~~volatile or non-volatile~~ computer ~~usable~~ readable storage medium having executable code, when executed by a computer, causes the computer to execute a ~~method~~ process for processing of a design that is based upon multiple programming languages, the design comprising a first language portion and a second language portion, the method comprising:

~~processing~~ performing a first process on the second language portion of the design to cause an interruption of processing for the first language portion of the design, wherein the processing for the first language portion is interrupted to process the second language portion, and the first language portion receives a plurality of requests for the processing for the first language portion from a programming interface when the first language portion is not interrupted;

determining whether there are one or more ~~waiting~~ of the plurality of requests, which are identified by and sent to a processing module to perform the first process on the second language portion by a programming interface, waiting for the processing of the first language portion and indicating a need for the processing module ~~of the second language portion~~ to call a request processing ~~function~~ module at the first language portion;

handling the one or more ~~waiting~~ of the plurality of requests, which are identified by and sent to the processing module by the programming interface, for the processing of the

first language portion by having the processing module ~~of the second language portion~~ call a request processing function at the first language portion that has been interrupted, at least one of the one or more ~~waiting~~ of the plurality of requests for the processing of the first language portion ~~causes the processing of the first language portion~~;

executing the request processing function at the first language portion to process the one or more ~~waiting~~ of the plurality of requests, wherein the act of executing the request processing function is performed by a processor; and

providing continued access to the first language portion while performing the first process on the second portion, by using the processing module, the second language portion; and

generating processing results based upon executing the request processing function and storing the processing results in a computer-readable storage medium or a storage device or displaying the processing results on a display apparatus.

30. (Currently Amended) A system for processing of a design that is based upon multiple programming languages, the design comprising a first language portion and a second language portion, the system comprising:

means for ~~processing~~ performing a first process on the second language portion of the design to cause an interruption of processing for the first language portion of the design, wherein the processing for the first language portion is interrupted to process the second language portion, and the first language portion receives a plurality of requests for the processing for the first language portion from a programming interface when the first language portion is not interrupted;

means for determining whether there are one or more ~~waiting~~ of the plurality of requests, which are identified by and sent to a processing module to perform the first process on the second language portion by a programming interface, waiting for the processing of the first language portion and indicating a need for the processing

module ~~of the second language portion~~ to call a request processing ~~function~~ module at the first language portion;

means for handling the one or more ~~waiting~~ of the plurality of requests, which are identified by and sent to the processing module by the programming interface, for the processing of the first language portion by having the processing module ~~of the second language portion~~ call a request processing function at the first language portion that has been interrupted, at least one of the one or more ~~waiting~~ of the plurality of requests for the processing of the first language portion ~~causes the processing of the first language portion~~;

means for executing the request processing function at the first language portion to process the one or more ~~waiting~~ of the plurality of requests and a computer readable storage medium or a storage device configured for storing processing results or a display apparatus configured for displaying the processing results ~~in a computer-readable medium~~, wherein the means for executing the request [[for]]processing function comprises a processor; and

means for generating processing results based upon executing the request processing function ~~and storing the processing results in a computer-readable medium~~.

31. (Previously Presented) The computer program product of claim 27 in which the simulator request accesses a portion of the HDL portion.

32. (Previously Presented) The computer program product of claim 27 in which the general programming language portion comprises C, C++, or SystemC code.

33. (Previously Presented New) The system of claim 28 in which the simulator request accesses a portion of the HDL portion.

34. (Previously Presented) The system of claim 28 in which the general programming language portion comprises C, C++, or SystemC code.

35. (Previously Presented) The computer program product of claim 29, wherein the request processing function is called by a gdb debugger.

36. (Previously Presented) The computer program product of claim 29 in which the one or more waiting requests are for debugging the first language portion.

37. (Previously Presented) The system of claim 30, wherein the request processing function is called by a gdb debugger.

38. (Previously Presented) The system of claim 30 in which the one or more waiting requests are for debugging the first language portion.